Handle 系统与域名系统互联互通机制：一种基于标记语言描述协议数据单元的实现 *

邹　慧 [1,2,3]，马　迪 [3,4]，王　伟 [2,3,4]，刘　阳 [5]，毛　伟 [2,3]，邵　晴 [3]

(1. 中国科学院计算机网络信息中心，北京 100190; 2. 中国科学院大学，北京 100049; 3. 北龙中网(北京)科技有限责任公司，北京 100190; 4. 互联网域名系统北京市工程研究中心，北京 101400; 5. 中国信息通信研究院，北京 100191)

摘　要：基于对理论和实践两个层面的认识，Handle 系统和域名系统将在未来很长一段时间内共存，而两种标志解析系统的解析协议和编码规则并不兼容，导致两者之间的数据空间无法共享，信息无法流通，降低了用户体验。因此，两种标志符解析系统的互联互通是当前亟需解决的一个问题。通过分析已有解决方案的利弊，发现协议数据单元与协议本身的分离可解决两个系统的解析协议和编码规则不兼容问题。利用这一分离机制，设计并实现了一种基于代理服务器的 Handle 系统与域名系统互联互通机制。实验结果表明，相对于传统客户端-服务器模式而言，该机制在不同应用场景下解析响应时间增量占比小，均在可接受范围内。

关键词：Handle 系统；域名系统；互联互通

中图分类号：TCP393.08　　doi: 10.3969/j.issn.1001-3695.2017.07.0680

Inter-operation mechanism for Handle system and domain name system: implementation based on markup language describing protocol data unit

Zou Hui[1, 2, 3], Ma Di[3, 4], Wang Wei[2, 3, 4], Liu Yang[5], Mao Wei[2, 3], Shao Qing[3]

(1. *Computer Network Information Center*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*; 2. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*; 3. *KNET Technologies*, *Beijing* 100190, *China*; 4. *Internet Domain Name System Beijing Engineering Research Center Ltd*, *Beijing* 101400, *China*; 5. *China Academy of Telecommunication Research*, *Beijing* 100191, *China*)

Abstract: Based on understanding of theory and practice, Handle System will coexist with the Domain Name System for a long time in the future, however, resolution protocols and identifier code rules of two identifier resolution system are incompatibility, leading to a barrier between data space of them. Therefore, it is urgent for us to resolve the connectivity problem of two identifier resolution system currently. By analyzing pros and cons of existing solutions, we found that the separation of protocol data unit and protocol can solve the incompatibility problem of resolution protocol and identifier code rule of two systems. Based on such a separation mechanism, inter-operation mechanism of Handle System and Domain Name System based on the proxy server was developed. Simulation experiments show that compared with traditional client-server ones, the response time of this new mechanism increases a little and is within the acceptable range.

Key Words: Handle system; domain name system; inter-operation

0　引言

DNS（domain name system，域名系统）[1]自 1983 年发明运行至今，因其技术的简单、实用，已成为多种应用程序的底层模块并提供定位功能，如 HTTP、FTP 和 Email 等。但是 DNS 的命名对象仅为互联网中的物理设备，不能够很好地满足互联网中数据资源对象的当前管理需求。URL（uniform resource Locator，统一资源定位符）[2] 是一种基于 DNS 技术为互联网上的各种类型数据资源对象提供较 DNS 更加细粒度命名服务的标志符。由于 URL 将“位置信息”嵌入数据资源对象的标志符

基金项目：工信部“2016 年智能制造综合标准化与新模式应用”专项资助项目

作者简介：邹慧（1991-），女，湖北宜昌人，博士研究生，主要研究方向为资源寻址定位技术、网络与信息安全（zouhui@knet.cn）；马迪（1984-），男，安徽六安人，博士，主要研究方向为互联网资源寻址定位技术、互联网基础资源管理技术；王伟（1977-），男，江苏镇江人，高工，硕导，博士，主要研究方向为可信网络、DNS 协议；刘阳（1986-），男，甘肃天水人，高工，博士，主要研究方向为工业互联网和物联网标识解析服务；毛伟（1968-），男，四川自贡人，研究员，博导，博士，主要研究方向为下一代互联网、资源寻址与定位；邵晴（1977-），男，湖南衡阳人，硕士.

中，导致数据资源对象发生迁移后其 URL 也需随之改变，否则将无法定位。

随着互联网数据资源对象的种类与数量越来越多，资源的重复存储和嵌套使用频次随之增加，存储空间的浪费现象严重，因此，对资源有效管理的需求日益增大。基于对以上需求以及现有技术缺陷的认识，Handle 技术[3]作为一种有效、可靠和可扩展的全球名字服务系统被提出。它的命名对象相较域名系统更加细化，在管理机制与服务模式上也有所不同，并且内嵌安全特性，而相较于 URL 则能为数据资源对象提供永久标志。但是，如果以 Handle 技术全面替换 DNS 技术，就目前而言在理论层面和实践层面均不现实：首先，应用层协议对于 Handle 解析模块的调用需要得到操作系统的支持，实现该技术对接还需要经历一个长期的过程；其次，虽然 Handle 系统已经在具体项目中得到应用，但由于应用领域的不同，还需要在实践中对其性能、效率和稳定性等方面进行实验和检验。由此看来，Handle 系统和域名系统因其自身的技术特点与应用范围将会在未来较长一段时间内共存，实现 Handle 系统与域名系统互联互通的必要性和重要性显而易见。

目前，学术界和工业界已有 Handle 系统与域名系统互联互通机制的相关研究。文献[4]基于 Handle 系统，通过名字空间的映射和数据记录格式的转换，将 DNS 系统的资源记录“翻译”后存储于 Handle 系统的一颗前缀子树下，以构建结合现有 DNS 技术和 Handle 技术特点的 Handle-DNS 名称服务系统，扩展并改善传统 DNS 服务系统的同时实现两个系统的互联互通。文献[5]提出了一种物联网异构标志符系统的编码和解析协议层面的互联互通机制，为每一种解析协议分配标准识别码并将其和解析协议和编码规则的映射关系信息存放于独立服务器中。而当前物联网编码解析系统使用的解析协议主要为 DNS 协议和 Handle 协议，该方案可视为 Handle 系统与 DNS 系统互联互通机制的“超集”。CNRI（Corporation for National Research Initiatives，美国国家研究推进机构）代表工业界率先提出解决方案，通过设置一个同时实现 HTTP 协议和 Handle 协议的 Web 服务器作为代理，从而实现两者的互联互通。用户在浏览器中输入携带 Handle 以及相关参数的 URL，Web 服务器代替用户与 Handle 系统直接交互以获得响应结果，并通过网页以 JSON 格式数据的形式予以展示。文献[4]默认客户端和服务器同时支持 Handle 协议和 DNS 协议，并且需要动态“翻译”DNS 数据空间，后续维护工作耗时耗力，扩展性不强，推广难度大。文献[5]要求客户端工作在“兼容解析模块”环境下，即客户端同时支持 DNS 协议和 Handle 协议，并不符合当前互联网终端设备协议的支持现状。CNRI 提出的代理方案能够很好地运行在“DNS 解析模块”环境下，即客户端仅需支持 DNS 协议即可。但该方案只能基于特定应用程序（浏览器）和信息表达方式（URL），用户能够表达的操作信息少且某些类型操作无法实现。另外，用户基于 HTTP 报文传输 JSON 格式数据的方式将会导致该方案在“受限环境”下无法得到应用，即终端设备并不支持 HTTP 协议和 JSON 标记语言的环境。

针对以上问题和不足，本文提出了一种以代理服务器作为“协议兼容协调者”来完成用户主机与 Handle 系统之间数据交换的机制。其基本原理是将 Handle 协议与 Handle 数据进行分离，即用户主机与代理服务器在数据传输之前协商双方所支持的传输协议和数据格式。通过将 Handle 请求和响应信息以该数据格式表示，并由协商的协议报文封装后传输，完成 Handle 数据在用户主机与 Handle 系统之间的交互。

1　背景

1.1　现状与未来

目前，用户终端操作系统仅支持 DNS 解析模块，Handle 技术作为一种新兴技术并非兼容存在，也即，当前处于“DNS 解析模块”环境下。未来，随着 Handle 应用领域的扩展，可以假设出厂的每一个用户终端天然地支持 DNS 解析模块和 Handle 解析模块，即未来将会处于“兼容解析模块”环境下，因此，Handle 系统与域名系统天然地实现互联互通。本文仅针对当前的“DNS 解析模块”环境，提出 Handle 系统与域名系统互联互通的一种解决方案，不仅为 Handle 系统与域名系统共存提供一个良好的信息共享环境，也为平滑过渡到“兼容解析模块”环境争取准备时间。

1.2　域名系统与 Handle 系统

域名系统作为域名与 IP 地址映射信息的一个分布式数据库，为人们提供方便访问互联网的服务。由于其自身技术特点，早已成为互联网基础协议并得到众多应用程序的广泛支持。域名系统经过长期运行考验，实践证明其大部分时间能够保持稳定运行，即使遭受攻击，也能在短时间范围内恢复得以正常提供服务，是互联网中不可缺少的一种寻址定位技术。

Handle 系统[6]于 1995 年由互联网之父 Robert Kahn 博士领导的 CNRI 开发，为互联网数据资源对象分配唯一标志符，并提供永久标志、动态解析和安全管理等服务。由于其强大的扩展性和兼容性，自发明以来，其受到的关注和可应用的领域范围与日俱增。它与域名系统类似，是一套集命名、注册与解析功能模块的完备标志解析系统，作为有其自身协议实现的另外一种寻址定位技术，与域名系统共存。因此，在信息高度共享的今天，域名系统与 Handle 系统数据空间的互联互通成为一个亟需解决的问题[7]。

1.3　互联互通

“互联互通”在本文范围内指 Handle 系统与域名系统之间实现互操作和互通，其中，互操作指用户主机能够向任一标志符解析体系发起任意类型标志符的查询请求；互通则指跨越不同标志符解析体系数据空间从而实现信息的开放、共享，换言之，数据空间并不针对特定类型协议的客户端可见。

2　概要设计

本方案通过架设服务器集群作为代理，完成“DNS 解析模

块”环境下的 Handle 信息获取。如图 1 所示，对于用户主机而言，首先，它需要识别应用程序发起的请求操作消息中携带的标志符类型，当类型为 Handle 时，通过主机配置协议或者带外方式配置的代理服务器位置信息完成定位，然后，用户主机与代理服务器完成传输协议（HTTP、CoAP、SIP 等）与数据格式（例如：JSON、XML、YAML 等）等参数的协商，最后，将 Handle 请求信息以协商的数据格式表示，并封装于传输协议报文内进行传输。因此，用户主机需要完成的操作内容可归纳为以下四种功能：标志符的识别、代理服务器的定位、数据格式的转换以及传输协议进程的调用。考虑到 Handle 请求信息的来源可能为多种应用程序，本机制以独立组件的方式实现上述功能，用户可根据自身需求决定是否为其使用的应用程序安装该组件。该方式既能满足用户主机应用程序差异的需求，也能满足应用程序应用场景差异的需求。

对于代理服务器而言，首先，它接受用户主机的传输协议报文并提取其内容，其次，由于用户掌控的 Handle 请求信息有限，代理服务器需要根据 Handle 报文添加相应缺失字段信息，并最终生成 Handle 请求报文。与之对应的响应过程，代理服务器接受 Handle 系统的 Handle 响应报文并提取用户可感知的内容，然后以协商的数据格式组织数据后封装于传输协议报文予以响应。



图 1　Handle over XXX 架构示意图

对于 Handle 请求/响应信息而言，它们在用户与 Handle 系统之间的传输过程中，由于经历不同的阶段而有其自身特点。每个阶段的实施主体、执行动作、数据状态以及是否添加/删除数据均不相同，数据的内容和格式随着到达的实施主体以及主体的执行动作发生相应的变化。

如表 1 所示，Handle 请求信息以用户为起点，经由用户主机、代理服务器，最终到达 Handle 系统。该过程可分为三个阶段，即用户到用户主机、用户主机到代理服务器和代理服务器到 Handle 系统。与之对应的实施主体分别为用户、用户主机和代理服务器，而数据状态为 Handle 请求信息、格式化封装数据以及 Handle 请求报文。如表 2 所示，Handle 响应信息以 Handle 系统为起点，经由代理服务器，由用户主机将最终 Handle 数据展示给用户。分为三个阶段，即 Handle 系统到代理服务器、代理服务器到用户主机和用户主机到用户。与之对应的实施主体分别为 Handle 系统、代理服务器和用户主机，而数据状态分别为 Handle 响应报文、格式化封装数据以及展示给用户的 Handle 响应信息，具有一定的灵活性。

3　详细设计

本文选用 HTTP 协议和 JSON 标记语言为例，分别为用户主机与代理服务器协商后选定的传输协议和数据格式，作为 Handle 数据的“传输隧道”和“描述工具”，详细说明本机制的实现细节。但是，在实际运行过程中，传输协议和标记语言并不局限于某一特定组合，而是根据用户主机和代理服务器的传输协议和标记语言支持情况而定。

表 1　Handle 请求信息传输阶段状态信息

实施主体	执行动作	数据状态	添加数据
用户	输入信息	Handle 请求信息	否
用户主机	格式化数据	格式化封装数据	否
代理服务器	适配 PDU	Handle 请求报文	是

表 2　Handle 响应报文传输阶段性状态信息

实施主体	执行动作	数据状态	删除数据
Handle 系统	解析响应	Handle 响应报文	否
代理服务器	适配 PDU&格式化 Handle 数据	格式化封装数据	是
用户主机	展示格式化 Handle 数据	Handle 响应信息	否

3.1　JSON 标记语言与 HTTP 协议

JSON 标记语言[8]是一种轻量级的数据交换格式，具有良好的可读性和便于快速编写特性，它适用于数据交换处理。作为 Handle 数据的“描述工具”，它将用户输入的 Handle 原始数据按照 Handle 协议报文的结构予以“描绘”，形成 JSON 格式的 Handle 数据。

HTTP 协议[9]作为 TCP/IP 协议族中的一员，它早已成为互联网基础协议并承载互联网上绝大多数流量，用户主机操作系统一般默认支持。它是一个面向内容的应用层协议，可携带数据对象的种类多样，因此，封装数据的格式选择空间大，得以简化协议兼容设计的细节问题，如标志符编码转换、协议数据单元适配以及数据记录格式转换等。另外，绝大多数防火墙、代理和控制设备开放 80 端口，HTTP 报文能够顺利穿透。作为 Handle 数据的“传输隧道”，它为格式化的封装数据提供传输途径[10]。

3.2　组件（用户主机）

应用程序作为用户输入 Handle 数据的入口，而组件负责处理无结构的 Handle 数据并将其组织成 JSON 格式数据。出于 PDU 适配效率的考虑，JSON 格式数据以 Handle 报文结构[11]为设计依据，Handle 报文由以下四个部分组成：封头（envelope）、头部（header）、实体（body）和凭证（credential），各司其职。封头用于指导 Handle 报文的传输，头部用来指导每一次 Handle 操作的具体内容，实体包含请求/响应数据内容，凭证则用于检测报文在传输过程中是否遭到窜改。

用户受其“Handle 知识”限制，无法在 Handle 报文生成前输入某些字段信息，例如封头部分和某些头部、凭证字段，而只能由实现 Handle 协议的实体（本文为代理服务器）予以填充。于是可知，Handle 报文内容的填充实体为用户和代理服务器，而 Handle 报文的生成实体仅为代理服务器。用户主机，即组件

能够感知用户输入的 Handle 请求/响应信息仅包含 Header、Body 和 Credential 的部分信息，分别对应 Handle 报文的头部、实体和凭证部分。每一个 Handle 请求/响应信息被定义为一个 JSON 对象。用户和代理服务器分别负责的字段如图 2 所示，灰色部分为用户负责字段，白色部分为代理服务器负责字段。

大版本 | 小版本 | 压缩 | 加密 | 分片
会话ID
请求ID
序列号
消息长度
操作码
响应码
权威 | 鉴别 | 加密 | 递归 | 缓存验证 | 连续 | 保持连接 | 大众可读 | 请求摘要
根信息版本号 | 递归次数
过期时间
实体长度
报文实体部分
凭证长度
凭证版本 | 保留 | 可选项
签名者信息<handle，index>
类型
签名信息

图 2　Handle 报文示意图

Handle 报文中由用户负责输入的字段信息如表 3 所示。

表 3　用户控制的 Handle 报文字段

部分	字段	含义
Header	OpCode	客户端发起的 Handle 操作类型
	ResponseCode	服务器对 Handle 操作的响应结果
	AT	请求操作必须由 Primary Site 处理
	CT	服务器对响应结果签名
	ENC	服务器使用预置会话密钥加密响应结果
	CA	缓存服务器对 Handle 服务器的响应结果进行验证
	KC	Handle 报文接受者返回应答后保持 TCP 连接
	PO	服务器仅需返回权限为 Public_Read 的数据记录
	RD	服务器在响应结果中包含 Handle 请求的摘要
Body		由 OpCode 和 ResponseCode 组合共同决定
Credential	Signer	签名者信息
	Type	签名内容的类型
	SignedInfo	签名内容的详细信息，包含三个字段：Length, DigestAlogrithm, SignedData，说明签名内容长度，签名使用的摘要算法和具体签名数据，第一个字段由代理服务器负责生成，后两个字段由用户负责输入

其中，对于 Handle 请求消息的 JSON 对象而言，除了 Body 部分的数据结构与具体操作类型有关，即由 OpCode 字段和 ResponseCode 字段共同决定以外(实际上由 OpCode 单独决定，所有 Handle 请求报文的 ResponseCode 字段均置 0)，Header 和 Credential 部分的数据结构相对固定，分别以一个 JSON 对象表示。图 3 和 4 给出了一个具体实例，直观地说明 Header 和 Credential 部分的 JSON 对象数据结构。

该具体实例表明用户发起一个 Handle 解析请求操作，并希望由 Primary Site 中的 Handle 服务器进行处理，并且仅返回权限为 Public_Read 的数据记录，缓存服务器需对响应数据进行验证。另外，用户对该 Handle 请求数据进行签名，以声明自身身份，实现代理服务器对用户的认证。其中，会话密钥保存在 Handle 值为 200/5555，索引值为 2 的数据记录中，签名算法为 SHA-A，并且签名内容为“shjdkwjhgdna2ixnmdmsnbdjhd”。


```
"Header"： {
        "OpCode"： "OC_RESOLUTION",
        "ResponseCode"： 0,
        "AuthoriTative"： 1,
        "CerTified"： 0,
        "ENcryption"： 0,
        "CacheAuthentication"： 1,
        "KeepConnection"： 0,
        "PublicOnly"： 1,
        "RequestDigest"： 0
}
```


图 3　JSON 格式化 Handle 请求/响应消息的 Header 部分示意图


```
"Credentials": {
        "version":0,
        "signerInfo": {
                "handle": "200/5555",
                "index": 2
        }
        "type": "HS_MAC",
        "signedInfo": {
                "digestAlg": "SHA-A",
                "signedData": {
                        "signature": "shjdkwjhgdna2ixnmdmsnbdjhd"
                        }
                }
}
```


图 4　JSON 格式化 Handle 请求消息/响应报文的 Credential 部分示意图

3.3　代理服务器

作为 Handle 消息传递的“中转加工站”，代理服务器通过与用户主机以及与 Handle 系统的两次交互过程共同完成一次 Handle 解析/管理操作。从 80 端口接收到 HTTP 报文后，代理服务器提取报文内的 JSON 格式数据，并根据图 2 所示 Handle 报文结构填充缺失字段信息，用以构建完整 Handle 请求报文。同理，在收到 Handle 响应报文后，代理服务器根据相关信息转换为 JSON 格式数据，并封装在 HTTP 报文内予以响应。在此过程中，与组件的功能相反，代理服务器需要丢弃用户并不感兴趣的信息内容，例如：用于指导 Handle 响应报文传输的字段内容。

对于 Handle 响应消息的 JSON 对象而言，除了 Body 部分的数据结构与具体操作类型有关，即由 OpCode 和 ResponseCode 字段共同决定以外，Header 和 Credential 部分的数据结构相对固定，分别以一个 JSON 对象表示，与 Handle 请求消息具有完全相同的数据结构，如图 3 和 4 所示。

由于代理服务器所处位置的特殊性，一方面，作为标志符解析过程中用户主机的代理人，代理服务器详细掌握每一次 Handle 请求操作内容和与之对应的 Handle 响应数据。该特性使得其可作为公共缓存服务器，保存每一次 Handle 请求/响应数据，以便迅速响应用户后续发起的针对同一前缀或者同一

Handle 的解析请求，提升用户体验的同时缓解 Handle 系统的压力。另一方面，作为用户主机与 Handle 系统的“沟通桥梁”，它将一条端到端的通信截断成两条连接，分别为用户主机到代理服务器和代理服务器到 Handle 系统。这两条连接并不孤立存在，而是共同合作完成一次 Handle 解析/管理操作，这种截断对于用户而言是透明存在的。因此，代理服务器为保证同属一条通信的两条连接正确匹配，需建立两条连接的状态信息映射表，包含会话 ID、会话密钥以及请求 ID 等内容。与此同时，由于端到端通信中引入了第三方实体，代理服务器作为一个“中间人”同时引入了安全隐患，安全问题不得不纳入考虑范围内，第五章对该问题进行了详细说明。

根据 RFC3652 Handle System Protocol (ver2.1) Specification 的内容可知，共有 14 种请求操作类型和 31 种响应结果类型，Handle 报文的实际内容由请求操作码（OpCode）和响应结果码（ResponseCode）共同决定。由于 Handle 报文中<OpCode，ResponseCode>组合不同而 Body 部分内容相同的情况存在，分析归纳得出 Handle 请求信息的 Body 部分共有 6 种数据结构，Handle 响应信息的 Body 部分共有 8 种数据结构。

一般而言，针对用户发起的 Handle 解析操作，Handle 系统返回的响应消息包含一个或者多个与该 Handle 关联的数据记录。Handle 系统中的数据记录分为自定义和预定义两种类型，预定义类型的数据记录一般用于 Handle 系统的管理操作，而自定义类型的数据记录可由组织机构在值为 0.NA/0.TYPE 的 Handle 前缀下进行注册。这两种类型的数据记录有一个共同的数据结构，可将其 JSON 格式化。图 5 给出了一个类型为 URL 数据记录的具体实例。


```
"record"： {
        "index": 1,
        "type":   "URL",
        "data": {},
        "ttl": 86400,
        "permission": ["PUBLIC_READ", "ADMIN_WRITE"],
        "timestamp": "2008-11-23T20:10:23Z",
        "reference": {
                "handle":"10/123",
                "index": 5
                }
}
```


图 5　数据记录的 JSON 对象示意图

数据记录包含七个字段，分别为索引、类型、数据、生存值、权限、时间戳和引用。每个字段均有其自身的语义和功能，详细信息可参见 RFC3651[12]，数据字段根据类型取值不同而具有不同的格式和内容，每一种数据记录的数据字段均有与之对应的 JSON 格式数据对象。

4　实验与分析

4.1　实验设计

本实验涉及的实体有以下三种类型：用户主机、代理服务器和 Handle 服务器，各实体对应的主机参数说明如表 4 所示。

考虑到代理服务器应用场景的不同，本实验设计了两组实验。第一组通过多进程方式模拟公共代理的服务环境；第二组通过多线程方式模拟私有代理的服务环境。为了评估本方案的优化效果，每组实验均有一个与之对应的对比实验，即用户主机与 Handle 系统直连的 Handle 解析过程。为了避免实验环境的不一致导致实验结果存在误差，两组实验中用户主机和 Handle 服务器的各参数均保持一致。

表 4　实验环境

主机	CPU	内存	操作系统
用户主机	32 Intel(R) Xeon(R) CPU E5-2640 v2@2.00GHz	128G	Linux Centos 7
代理服务器	32 Intel(R) Xeon(R) CPU E5-2670 0@2.6GHz	64G	Linux Centos 7
Handle 服务器	32 Intel(R) Xeon(R) CPU E5-2670 0@2.6GHz	64G	Linux Centos 7

4.2　实验结果

考虑到 Handle 系统作为标志解析系统，其主要功能在于提供 Handle 标志符的解析查询服务，本实验仅针对解析操作并从响应时间这一个维度来对比分析基于代理服务器机制与传统直连机制。

由于私有代理的 Handle 并发解析查询量在一定时间范围内相较于公有代理少，本实验中将私有代理环境下和公有代理环境下的最大解析查询量分别设置为 10 000 次和 50 000 次。

4.2.1 公共代理

公共代理对其服务对象并不加以任何限制条件，因此其接受的解析请求中的 Handle 具有随机性，并不具有强关联性。由于进程是资源分配的最小单元，拥有独立的数据空间、CPU 等，因此，多进程方式可模拟公共代理的服务环境，即各解析请求之间相互独立，自行完成 GHR（Global Handle Registry，全球 Handle 注册局）——>LHS（local Handle service，本地 Handle 服务）——>Handle 的完整查询流程。

本实验设置 20 个进程，用于模拟用户的解析请求行为，对比直连 Handle 解析过程和基于代理服务器的 Handle 解析过程在并发解析情形下的时间效率。计算从第一个解析请求到最后一个响应结果的总耗时，从而得出各机制的平均解析响应时间。各组实验分别进行九次，每一次设置不同数量的解析请求且依次递增，结果如图 6 所示。

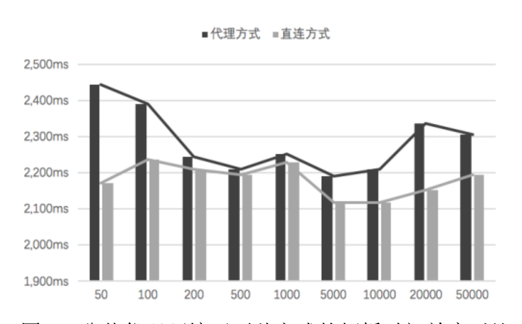


图 6　公共代理环境下两种方式的解析时间效率对比

第一列为基于代理服务器的 Handle 解析过程（代理方式），第二列为基于 C/S 方式的 Handle 解析过程（直连方式）。

通过图 6 可知，公有代理环境下代理方式和直连方式的平均解析响应时间均在一定范围内波动，且代理方式的平均解析响应时间普遍高于直连方式的平均解析响应时间。但是，代理方式的平均解析响应时间增量相较于直连方式的平均解析响应时间而言占比小，可忽略不计。

4.2.2 私有代理

由于私有代理的服务对象一般局限于一个组织内部，例如工厂、物流公司等，因此解析的 Handle 具有一定的前缀或者访问强关联性。由于一个进程内的多线程共享资源（CPU、数据空间），因此，多线程方式可模拟私有代理的服务环境，即解析请求之间具有一定的依赖性。这种情况下，后续解析请求可复用第一个解析请求从 GHR 获取的 LHS 信息，省略了查询 GHR 的步骤。

本实验设置 20 个线程，用于模拟用户解析请求行为，对比直连 Handle 解析过程和基于代理服务器解析过程在并发解析情形下的时间效率。计算从第二个解析请求到最后一个响应结果的总耗时，从而得出各过程的平均解析响应时间。由于解析请求之间的依赖性，以及首个解析请求和后续解析请求响应时间的差异性，因此，本实验选用的测量参数为排除首个解析请求操作的平均解析响应时间。各组实验分别进行七次，每一次设置不同数量的解析请求且以依次递增，结果如图 7 所示。

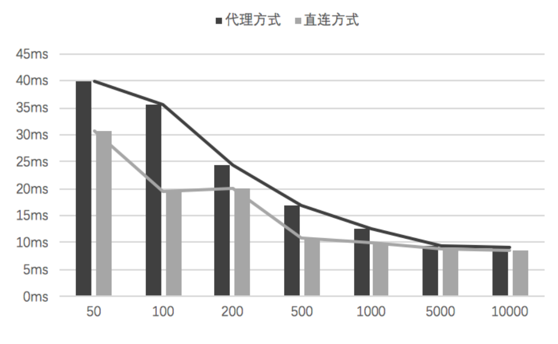


图 7　私有代理环境下两种方式的解析时间效率对比

第一列和第二列的含义与图 6 相同。

通过图 7 可知，私有代理环境下代理方式的平均响应时间普遍高于直连方式的平均响应时间，但是，随着解析查询量的增加，代理方式的平均解析响应时间逐渐趋向于直连方式的平均解析响应时间。另外，通过图 6 和 7 可看出，在公有代理环境下，由于解析的 Handle 之间关联性小，相较于私有代理环境下随机性更大，因此，公有代理环境下两种方式的平均解析响应时间主要用于向 GHR 查询 Handle 的 LHS 信息的过程中。

5　安全与性能

Handle 系统具有内嵌的安全特性，提供端到端的加密与签名认证服务，详细来说，即客户端与服务器可对传输数据进行签名和加密。其中，签名用于通信双方认证对方的身份信息并提供不可否认性服务，而加密则用于为传输数据提供私密性和完整性服务。代理服务器由于功能特性决定了其“中间人”的特殊角色性质，即它需要解读并重构用户发起的 Handle 查询数据和 Handle 系统返回的响应报文。由于端到端的加密与签名仅面向协议数据单元，换言之，代理服务器成为一个可信任的“中间人”，能够嗅探到用户与 Handle 系统通信的全部数据内容，因此在本方案中，端到端的安全特性无法得到保障，代理服务器成为一个潜在的安全隐患。

根据 RFC 3651 中 4.2.2 小节的说明[10] 可知，代理服务器并不属于 Handle 系统的管理或者验证范畴，即用户并不验证从代理服务器返回的 Handle 响应数据。用户是否信任代理服务器返回的 Handle 响应数据取决于用户是否与代理服务器建立了信任关系，而信任关系由用户根据自身策略决定是否建立，信任的主动权掌握在用户手中。另外，当与代理服务器建立信任关系后，用户可以依赖代理服务器对从 Handle 系统返回的应答数据进行验证。

代理服务器根据服务对象与其的关系可以分为公共代理和私有代理。当使用不同类型的代理服务器时，用户对它们的信任度是存在差异的。公共代理类似于域名系统中的公共递归解析服务器，例如 Google 公司的 IP 地址为 8.8.8.8 的递归解析服务器，它对服务对象并不施加限制条件，一般独立于任何组织机构。由于用户与公共代理并不具有任何服务约束关系，因此，当用户使用公共代理时存在信任关系建立问题。用户默认不信任公共代理，需要根据自身策略决定是否将一个公共代理列入信任列表以供后续使用，需要考虑的因素包括但不限于如公共代理的声誉、地理位置、维护者信息等等。此时，用户需要在安全性和可用性之间做权衡考量：当决定信任并使用公共代理时则存在潜在的安全隐患；当决定不信任公共代理时则无须牺牲安全代价，但可用性无法得到满足。

私有代理类似于域名系统中一个组织机构内部自行搭建的递归解析服务器，服务对象仅面向组织内部成员，且与服务对象共同存在于组织内部构成的信任域中。正常情况下，私有代理为一个组织机构内部专有的服务器，它默认不进行恶意操作且不对组织机构内部其他成员构成安全威胁。在工业领域范畴内普遍存在的情况为一个组织机构形成一个应用闭环，同时也构成一个信任域，即组织机构内部各实体默认相互信任。因此，当用户使用私有代理时并不存在信任关系建立问题，组织机构内部成员默认信任私有代理，信任关系在初始化时即建立成功，不需要用户自行配置信任列表。

另外，代理服务器作为互联互通系统的关键“连接纽带”，需要处理的事务繁多，如报文内容的提取、协议数据单元的适配、映射信息的维护等等，容易成为整个系统的性能瓶颈。同时，代理服务器也容易成为攻击的对象，如 DDos 攻击，如果攻击一旦成功，轻则应答延时增大，重则无法正常提供服务。因此，出于对安全和性能的考量，可通过设置代理服务器集群，结合负载均衡算法或者 BGP+Anycast 等方式可以避免代理服

务器的负担过重，同时防止由于攻击导致的单点故障。

6　结束语

针对 Handle 系统与域名系统由于解析协议不兼容而无法实现数据空间共享的问题，本文首先分析了针对该问题提出的三种解决方案的利弊，并设计了一种基于代理服务器的协议与数据分离机制，并给出了相应的实现细节。本文还分析了代理服务器的应用场景以及相关的安全和性能问题，并针对不同类型的代理服务器给出了相应的实验方案。实验结果表明，在公共代理环境下，本方案与直连方式的平均响应时间相差不大，可忽略不计；在私有代理环境下，本方案在小批量的解析情形下，平均解析响应时间增量在可接受范围内，而随着解析查询量的增加，本方案的平均解析响应时间已表现出逐渐趋向于直连方式平均解析响应时间的趋势，且增量仅为几毫秒。

随着 Handle 标志解析系统在工业领域的进一步应用，并基于对工业领域内组织机构以及解析操作特征的认识——组织结构自身构成一个应用闭环且物品的查询解析具有批量操作性。因此，该方案在保证域名系统与 Handle 系统互联互通的同时并未显著降低解析操作的时间效率，具有良好的应用场景。

参考文献：

[1] Mokapetris P. Domain names: concepts and facilities, RFC1034 [R]. 1987.
[2] Berners-Lee T, Masinter L, McCahill M. Uniform resource Locators (URL) , RFC 1738 [R]. 1994.
[3] Sun S, Lannom L, Boesch B. Handle system overview, RFC 3650 [R]. 2003.
[4] 王峰，孙洵，毛伟，等. Handle-DNS 名字服务系统的设计与实现 [J]. 微电子学与计算机, 2004, 21 (1): 39-44.
[5] 孔宁，沈烁，刘冰，等. 一种物联网异构标志解析方法与系统：中国, 2013104076132 [P]. 2013-09-09.
[6] 郭晓峰，孙洵. Handle 系统的发展及应用 [J]. 数字图书馆论坛, 2013, (8): 18-24.
[7] Kahn R, Maffei A. Terminology and use cases for interoperability of identifier resolution systems [EB//OL]. draft-kahn-dsii-id-res-sys-00, 2012.
[8] Bray T. The JavaScript object notation (JSON) data interchange format, RFC7159 [R]. 2014.
[9] Fielding R, Gettys J, Mogul J, et al. Hypertext transfer protocol-HTTP//1. 1, RFC2616 [R]. 1996.
[10] Popa L, Ghodsi A, Stoica I. HTTP as the narrow waist of the future Internet [C]// Proc of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. 2010.
[11] Sun S, Reilly S, Lannom L, et al. Handle system protocol (ver 2. 1) specification, RFC 3652 [R]. 2003.
[12] Sun S, Reilly S, Lannom L. Handle system namespace and service definition, RFC 3651 [R]. 2003.